U.S. NAVAL ACADEMY
COMPUTER SCIENCE DEPARTMENT
TECHNICAL REPORT

Internet Protocol Security (IPSEC): Testing and Implications on IPv4 and IPv6 Networks

Domagalski, Joshua E.

USNA-CS-TR-2008-02

August 27, 2008

USNA Computer Science Dept. ◦ 572M Holloway Rd Stop 9F ◦ Annapolis, MD 21403

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
27 AUG 2008		**00-00-2008 to 00-00-2008**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Internet Protocol Security (IPSEC): Testing and Implications on IPv4 and IPv6 Networks	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **U.S. Naval Academy,Computer Science Department,572M Holloway Rd Stop 9F,Annapolis,MD,21403**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**24**	

U.S. NAVAL ACADEMY
COMPUTER SCIENCE DEPARTMENT
TECHNICAL REPORT

Internet Protocol Security (IPSec): Testing and Implications on IPv4 and IPv6 Networks

Domagalski, Joshua E

USNA-CS

April 28, 2008

Computer Science Department
SI495B: Research Project Report
Spring AY08

Internet Protocol Security (IPSec): Testing and Implications on IPv4 and IPv6 Networks

by

Midshipman Joshua Domagalski, 081812

United States Naval Academy
Annapolis, MD

_____

_____
Date

Certification of Faculty Mentor•s Approval

Assistant Professor Patrick Vincent
Department of Computer Science

_____

_____
Date

Assistant Professor Thomas Augustine
Department of Computer Science

_____

_____
Date

Department Chair Endorsement

Captain Thomas Logue
Chair, Department of Computer Science

_____

_____
Date

Executive Summary

The research study, *Internet Protocol Security (IPSec): Testing and Implications on IPv4 and IPv6 Networks,* was conducted at the United States Naval Academy (USNA) with the goal of developing, employing, testing, and analyzing the IPSec protocol over IPv4, IPv6, and IPv4/IPv6 networks. Specifically, the study focused on the IPSec algorithm and the implications for its use on IPv6 and IPv4/IPv6 networks.

The Office of Management and Budget (OMB) has mandated the complete conversion from IPv4 to IPv6 by 2008. In an effort to undertake this conversion, the Department of Defense (DOD) has directed the Defense Information Systems Agency (DISA) to devise a roadmap for the conversion to IPv6. The most important reason for the conversion to IPv6 is the interest in improving network security: IPv6 mandates the use of Internet Protocol Security (IPSec) in the IPv6 stack. Though from cursory review, IPv6 (utilizing IPSec) appears to be more secure than IPv4, there are many considerations revealed in this study which render IPv6 security enhancements more complex to obtain than originally thought.

The goals we established from the onset of the research study were threefold: 1) to test and analyze the implementation of IPSec over an IPv4 network, 2) to test and analyze the implementation of IPSec over an IPv6 network, and 3) to test and analyze the implementation of IPSec over an IPv4/IPv6 network. These were to be tested using common, current operating systems to include Windows XP SP2 and SUSE 10.3 Linux. Of particular interest, we noted that IPSec poses many problems with Network Address Translation and in handling the incompatibility between IPv4 and IPv6 packets (due to differences in the two protocol•s packet headers). In addition, we found that it is vital to understand the theory, implementation, and application of IPSec for the purposes of authentication and encryption.

By the conclusion of the study, we were able to create a fully functional IPv4 protocol that implemented IPSec for file transfers and ICMP packets. Although we attempted to implement IPSec on an IPv6 network, we found that neither Windows XP Service Pack 2 nor Novell•s SUSE 10.3 fully support IPSec on IPv6 networks. In addition, many of the limitations that hinder the implementation of IPSec are not caused by faults in the protocol itself, but rather by problems with the applications utilizing the protocol.

Our research provides recommendations for future participation and study in this crucial area of network security.

1. Introduction

The Internet has revolutionized communication, enabling academic institutions, research institutes, private companies, government agencies and individuals to have worldwide public access to interconnected computer networks that transmit data using the standard Internet Protocol (IP). This communication phenomenon has rendered instant access to an exponentially growing amount of information, facilitating what has fast become a global economy. No longer exclusively limited to the transmission of rudimentary data, there is an increasing demand to facilitate more sophisticated real-time communication forms such as voice and video. Along with the inherent benefits and advancements of such a revolutionary communication network, attention has focused on the ever-growing need for information security in the both the private and public sectors. It is with regard to this critical need that the basic Internet Protocol has been found lacking; it lacks an inherent ability to provide an all-inclusive, end-to-end security solution in the network layer of the Open Systems Interconnection (OSI) basic reference model.

Due to the inherent nature of the Internet, where streams of data merge and flow through different routers and links before reaching their final destination, it becomes imperative to use cryptography? the study and practice of technologically hiding information? as the means for obtaining security and privacy. This security attempt has commonly been implemented in the application layer of the OSI model. However, if properly implemented in the application layer, every encryption becomes application-specific and, as a result, a very heavy burden is placed on the system administrator. In addition, many encryption schemes can fall prey to man-in-the-middle attacks or to the requirement that symmetric keys be exchanged over insecure channels. To avoid these problems, digital certificates and public-key cryptography have been extensively utilized, but public key cryptography is very slow even with modern processors.

Thus, in an attempt to provide an end-to-end security solution governed by the network layer, the IP Security (IPSec) suite was implemented. IPSec seamlessly merges several protocols and integrates with both IPv4 and IPv6 to provide source authentication, integrity, confidentiality, and replay-attack protection (and can also be used to construct Virtual Private Networks (VPNs), thus connecting distinct networks into a single larger one) without the inherent limitations of utilizing the application layer of the OSI model. In addition, ICMP, TCP and UDP protocols all work with IPSec.

2. IPSec

IPSec is the combination of many different protocols into a comprehensive architecture. Currently, IPSec comprises the following protocols: AH (Authentication Header), ESP (Encapsulating Security Payload), IKE (Internet Key Exchange), ISAKMP/Oakley (Internet Security Association and Key Management Protocol), and associated transforms. However, from a practical application standpoint, IPSec is normally applied to IP packets in the form of Authentication Headers or Encapsulating Security Payloads. We briefly describe the protocols used in IPSec.

2.1 AH: Authentication Headers

AH?as the name implies? is used to authenticate IP packets. Thus it only guarantees that the received packet was not modified while in transit, that it was not sent by a third party (packet spoofing), and, if implemented, that it is a new, non-replayed packet. As such, AH does not provide any encryption capabilities and thus no confidentiality. AH does not mandate the underlying authenticating algorithm that must be used, but rather relies upon two other protocols, ISAKMP and IKE, to determine the appropriate algorithm. Two underlying authentication algorithms that all users must support are HMAC-SHA-96 (which is an acronym for Hashed Message Authentication Code-Secure Hash Algorithm-96) and HMAC-MD5-96 (Hashed Message Authentication Code-Message Digest 5-96). Due to the processing power consumption and slowness of public key authentication methods, RSA and Digital Signature Standard (DSS) are currently not implemented in or defined for use with AH.

AH can be implemented in *transport mode* or *tunnel mode*. When implemented with transport mode, the AH header immediately follows the IP header but comes before upper-layer protocol headers. This implementation provides basic, end-to-end communication protection. When implemented with tunnel mode, the AH encapsulates the protected packet datagram. Thus the original packet maintains the same IP addressing while an additional IP header is added before the AH header for tunnel addressing. The following provides an example of an original IP datagram, an AH transport datagram, and an AH tunnel datagram:

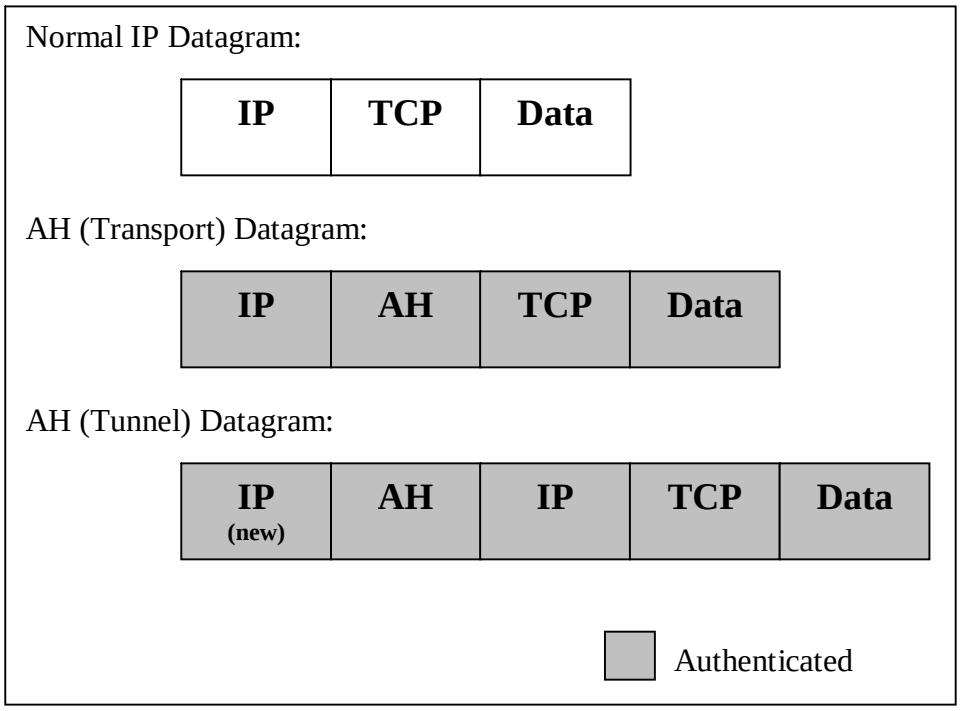

Figure 1: AH Datagram Comparison

When AH is implemented, an outbound Security Association (SA) is created either by IKE or by manual configuration. To provide the ability to prevent replay attacks, a sequence number counter is initialized to zero. Prior to the construction of each AH header, the SA is incremented, thus ensuring an increasing, unique, nonzero,

monotonically increasing number. As AH only extends protection to the immutable fields of the outer IP header, all mutable fields must be •zeroed out• before computing the Integrity Check Value (ICV). The ICV is calculated by passing both the key obtained from the SA as well as the entire IP packet (with the mutable fields zeroed out) to the algorithm defined in the SA. The ICV value is then copied into the authentication data field of the AH, and all of the mutable fields are then filled.

The authenticating of immutable fields in an attempt to provide authenticity for the packet necessitates that the source IP address and destination IP address be included in the ICV. However, this provides a unique problem concerning Network Address Translation (NAT) ? a network setup commonly seen. Because NAT forwards packets from a gateway to host computers, and thus requires the changing of the destination IP address, the authentication capability of AH collapses in an end-to-end AH transport mode setup involving NAT.

2.2 ESP: Encapsulating Security Payload

Encapsulating Security Payload, however, provides both authentication as well as confidentiality. As was the case with AH, anti-replay services are provided, though not mandated.

As the ESP header is an IPSec header, it contains an SPI field which is an arbitrary number selected during an IKE exchange. This SPI value is not encrypted, but rather is only authenticated and is used to identify the encryption algorithm and key needed to decrypt the packet. All actual data that is encrypted, or protected by ESP, is held within the payload data field. Due to the nature of block cipher encryptions, padding is used in ESP to maintain boundaries, thus providing flexibility for different encryption schemes. If ESP is being implemented in transport mode, then the ESP header will be placed between the upper-level and the IP headers. In other words, when using IPv4 in transport mode, the ESP header will be placed immediately after the IP header. However, if using IPv6 in transport mode, the ESP header is placed after the extension headers. For tunnel mode, on the other hand, the ESP header is •prepended• to the IP packet ? the IP header contains the source address of the device applying the ESP and the destination address is obtained from the SA. The following provides an example of an original IP datagram, an ESP transport datagram, and an ESP tunnel datagram:

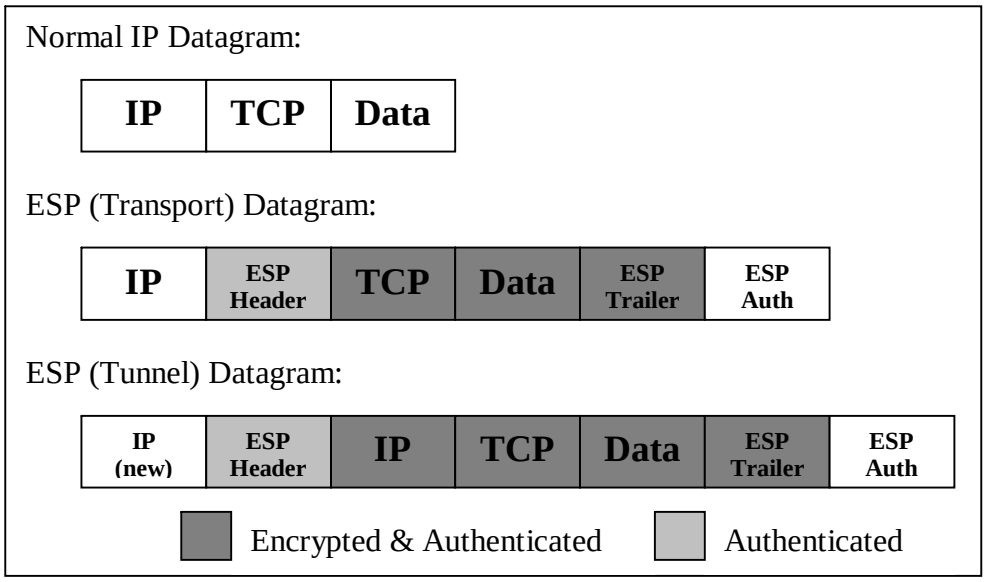

Figure 2: ESP Datagram Comparison

2.3 ISAKMP: Internet Security Association and Key Management Protocol

In order for any encryption to occur between two different entities, at least two phases must occur. The first phase is communication as to the type and method of implementation of the encryption. This normally establishes what algorithm to use, what key to use, how to authenticate each other, and what traffic needs to be encrypted. The second phase is the actual encryption and relaying of encrypted information.

IPSec•s ISAKMP (Internet Security Association and Key Management Protocol) governs how two peers communicate, how messages are constructed, and what states the peers use to secure their communication. Although the ISAKMP protocol defines and provides a means of authentication, information for key exchanges, and the negotiation of security services, it does not define either how particular keys are exchanged or the necessary security associations. ISAKMP denotes two exchange phases. The first exchange phase establishes an SA (this is not the same as an IPSec SA) for the communication. This relays information on how the peers will authenticate each other to provide protection to phase two. The second phase establishes SAs for other protocols. Once phase two ends, the ISAKMP process is destroyed.

The first step of any ISAKMP exchange is the generation and exchanging of cookies. These cookies are 8 bit pseudo-random numbers generated by each entity. Each cookie is unique and is normally constructed using the *Photuris* key exchange: a hash of the entity•s IP address, port, protocol and some timestamp and is referred to as the Initiator Cookie (IC) and the Responder Cookie (RC). This key exchange is done prior to any intensive operations (i.e. Diffie-Hellman exponentiation). This allows for the dropping of bogus ISAKMP messages that a denial-of-service attacker would use.

When establishing a shared SA, a flexible parsing of SA, proposal, and transform payloads must be allowed. In order to identify the communications and respective SA, a Message ID (MID) is given to the communication after the initial response. In addition, after both cookies are transmitted, they are concatenated to form the Security Parameters

Index (SPI) that will identify the SA for that communication. The following table provides a breakdown of what is present during each respective phase of the communication:

Table 1: ISAKMP Communications (RFC 2408)

		IC	RC	MID	SPI
1)	Start ISAKMP SA Negotiation	X	0	0	0
2)	Response ISAKMP SA Negotiation	X	X	0	0
3)	Initialize other SA Negotiation	X	X	X	X
4)	Respond other SA Negotiation	X	X	X	X
5)	Other (KE, ID, etc)	X	X	X/0	N/A
6)	Security Protocol (ESP, AH)	N/A	N/A	N/A	X

2.4 IKE: Internet Key Exchange

As ISAKMP does not conduct an actual key exchange, the Internet Key Exchange (IKE) protocol augments the IPSec suite thus allowing for an authenticated key and agreed-upon security services between two hosts. Like ISAKMP, IKE is a two-phase exchange: the first phase establishing the IKE SA and the second phase using the SA to negotiate further security associations for IPSec. One unique aspect of IKE, however, is that it is not limited to IPSec alone, but is rather a generic protocol that can be used in routing protocols (RIP and OSPF).

IKE denotes two types of phase one exchanges, one type of phase two exchange, and two other exchanges. The two types of phase one exchanges are known as •main mode• and •aggressive mode.• Phase two is then known as •Quick Mode•: a phase where all other security services other than IKE are then negotiated ? including IPSec. The other two exchanges are •informational exchanges•, used for status information and communication errors, and •new group exchanges•, used to negotiate the use of a new Diffie-Hellman group.

Table 2: IKE Modes and Phases

<table>
<tr><th colspan="2">Phase</th></tr>
<tr><th>IKE Phase 1</th><th>IKE Phase 2</th></tr>
<tr><td>Main Mode
6 messages, identity protected</td><td rowspan="2">Quick Mode
3 messages, establishes parameters for ESP,AH, SHA, MD5 and SA lifetime and session keys</td></tr>
<tr><td>Aggressive Mode
3 messages, no identity protection</td></tr>
</table>

All of the security parameters used by IKE are referred to as a •protection suite• ? encryption and hash algorithm, authentication method, and Diffie-Hellman group. These protection suites are negotiated between hosts by exchanging ISAKMP SA payloads ? all contained in transform payloads. Most hash algorithms use the HMAC form in the form

of a pseudo-random function (PRF). For more information about the Diffie-Hellman, see Appendix A.

Central to the parameters established for the Diffie-Hellman exchange is the Diffie-Hellman group being used as it determines the key created and exchanged. These are predefined groups, of which there are currently five:

1. a Modular Exponentiation (MODP) group with a 768-bit modulus
2. a MODP group with a 1024-bit modulus
3. an Elliptic Curve Group over GF[2^n] (EC2N) group with a 155-bit field size
4. an EC2N group with a 185-bit field size
5. a MODP group with a 1680-bit modulus

Algorithmically, it is considered that groups 1 and 3 and groups 2 and 4 provide similar levels of security. However, there is no elliptic curve analog to the modulus prime group 5. After the Diffie-Hellman group is determined, the authentication utilizes one of four primary means for authentication for IPSec: preshared keys, exchanging encrypted nonces, digital signatures using the Digital Signature Algorithm, and the Rivest-Shamir-Adelman (RSA) algorithm. For more information about the implementation of these algorithms, see Appendix B.

3. Research Study

A primary purpose of this study was to test the full scale implementation of IPSec•s Authentication Headers and Encapsulating Security Payloads. Since we wished to test the implications of IPSec upon NAT, the use of a router was imperative. In addition, the primary OS chosen was Windows XP due to the wide commercial availability and support, in addition to the reliance of the Department of Defense upon the Windows XP OS. As a Linux variant, we tested SUSE 10.3 from Novell.

3.1 Initial Setup and Configuration

Because we wanted to provide the ability to test three possible variations of IPSec implementation? Ipv4 only, IPv6-only, and IPv4-IPv6? we set up each of the four computers as a VLAN on our Cisco 3845 router. On each computer we installed VMware running, in addition to the host operating system, Windows XP SP2 and SUSE 10.3. Initially, all computers were setup as IPv4 only. Thus our network was designed as the following diagram depicts:

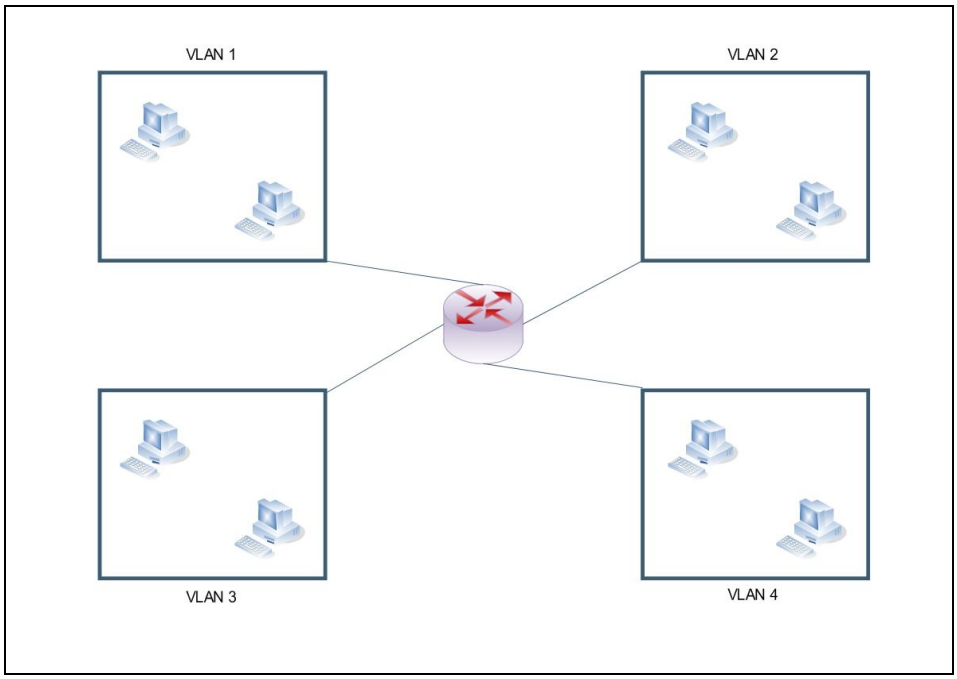

Figure 3: Network Layout

After configuring the router for Network Address Translation (NAT) on each VLAN, we installed Wireshark on each host computer. The following lists the manner in which we organized the machines:

Red-King (WIN XP SP2):	**192.168.4.1/24**
Black-King (SUSE 10.3):	**192.168.4.2/24**
Red-Queen (WIN XP SP2):	**192.168.3.1/24**
Black-Queen (SUSE 10.3):	**192.168.3.2/24**
White-King (WIN XP SP2):	**192.168.2.1/24**
Blue-King (SUSE 10.3):	**192.168.2.2/24**
White-Queen (WIN XP SP2):	**192.168.1.1/24**
Blue-Queen (SUSE 10.3):	**192.168.1.2/24**

We also enabled the IPv6 package on the Windows XP SP2 machines for the following IPv6 link-local addresses:

Red-King (WIN XP SP2):	**fe80::20c:29ff:fede:7734**
Red-Queen (WIN XP SP2):	**fe80::20c:29ff:fe0e:e216**
White-King (WIN XP SP2):	**fe80::20c:29ff:fe8b:76be**
White-Queen (WIN XP SP2):	**fe80::20c:29ff:fe2e:ef43**

After setting up the IP addresses, we conducted basic pinging and file transfer using ftp to ensure that the network was properly configured.

3.2 IPSec Policy Configuration

We started by configuring the Windows boxes to use IPSec ESPs for all traffic. By accessing the secpol.msc console, we were able to select and add rules to require security for all traffic, both incoming and outgoing. Windows XP offers three options when configuring for IPSec: Kerberos, Certificates, and Pre-shared Key. Due to the nature of our setup, we used the pre-shared key •usna• on all of the Windows machines. The encryption algorithms allowed were DES, 3DES, and AES. In an effort to lessen the complexity of the encryption in the hopes of mathematically proving how IPSec was being implemented, we utilized DES as our primary, or first transform. We selected MD5 for the HMAC-hash. We also selected the •Low• 768 bit Diffie-Hellman group, without perfect forward secrecy.

By setting the policy to •require• authentication, every protocol including ICMP was affected, thus resulting in pinging problems between the Windows and Linux boxes. Of note is that the authentication headers are not affected by internal NAT schemes between VLANs. The only situation where the problem was made manifest was in the translation between external and internal 1:1 NAT schemes.

3.3 IPSec File Transfer and Packet Sniffing Analysis

In order to see the ISAKMP and IKE protocols being implemented, we setup a test FTP Server using XLight FTP Server on White-Queen. We then created a test text document with •United States Naval Academy• written inside. Prior to any connection, we turned off all of the IPSec policies on White-Queen and Red-King and initiated Wireshark on the two host machines of White-Queen and Red-King. After connecting, we then transferred the file. The Wireshark output proved that the entire transaction had, as we intended, been entirely clear-text.

We then configured the IPSec policy for the two computers to require Authentication only and no encryption. After conducting the same file transfer experiment again, we noted the implementation of the ISAKMP protocol and the 6 messages of Main Mode followed by the 3 messages of Quick mode. The first packet gave the following information regarding the Initiator Cookie, the exchange type, and the next payload to expect:

```
Internet Security Association and Key Management Protocol
   Initiator cookie: FEDFDC8BB4E8189E
   Responder cookie: 0000000000000000
   Next payload: Security Association (1)
   Version: 1.0
   Exchange type: Identity Protection (Main Mode) (2)
   Flags: 0x00
      .... ...0 = Not encrypted
      .... ..0. = No commit
      .... .0.. = No authentication
   Message ID: 0x00000000
   Length: 276
```

Immediately following the above snippet, the transforms of the Initiator were listed as per the following example:

```
Proposal payload # 1
        Next payload: NONE (0)
        Payload length: 152
        Proposal number: 1
        Protocol ID: ISAKMP (1)
        SPI Size: 0
        Proposal transforms: 4
        Transform payload # 1
           Next payload: Transform (3)
           Payload length: 36
           Transform number: 1
           Transform ID: KEY_IKE (1)
           Encryption-Algorithm (1): DES-CBC (1)
           Hash-Algorithm (2): MD5 (1)
           Group-Description (4): Default 768-bit MODP group (1)
           Authentication-Method (3): PSK (1)
           Life-Type (11): Seconds (1)
           Life-Duration (12): Duration-Value (60)
```

For the full packet, four transform payloads were listed in the order of precedence as listed by the SA created on the system. The following response included the following Responder Cookie:

```
Internet Security Association and Key Management Protocol
   Initiator cookie: FEDFDC8BB4E8189E
   Responder cookie: F019425F61E3FB68
   Next payload: Security Association (1)
   Version: 1.0
   Exchange type: Identity Protection (Main Mode) (2)
   Flags: 0x00
      .... ...0 = Not encrypted
      .... ..0. = No commit
      .... .0.. = No authentication
   Message ID: 0x00000000
   Length: 148
```

The Responder also replied with the first payload as both computers had been configured for the same hierarchy of SA transforms. As the transform finished, the next packet included the nonce exchange from the Initiator as follows:

```
Internet Security Association and Key Management Protocol
   Initiator cookie: FEDFDC8BB4E8189E
   Responder cookie: F019425F61E3FB68
   Next payload: Key Exchange (4)
   Version: 1.0
   Exchange type: Identity Protection (Main Mode) (2)
   Flags: 0x00
      .... ...0 = Not encrypted
      .... ..0. = No commit
      .... .0.. = No authentication
   Message ID: 0x00000000
   Length: 192
   Key Exchange payload
      Next payload: Nonce (10)
      Payload length: 100
      Key Exchange Data (96 bytes / 768 bits)
   Nonce payload
      Next payload: NAT-D (draft-ietf-ipsec-nat-t-ike-01 to 03) (130)
      Payload length: 24
      Nonce Data
   NAT-D (draft-ietf-ipsec-nat-t-ike-01 to 03) payload
      Next payload: NAT-D (draft-ietf-ipsec-nat-t-ike-01 to 03) (130)
      Payload length: 20
      Hash of address and port: 3091D1A5834CD9B8B59054BA994F5389
   NAT-D (draft-ietf-ipsec-nat-t-ike-01 to 03) payload
      Next payload: NONE (0)
      Payload length: 20
      Hash of address and port: 03C4F5874FBB9FA58ABA4B19906DC59E
```

This packet was reciprocated by the Responder with the following nonce transfer:

```
Internet Security Association and Key Management Protocol
   Initiator cookie: FEDFDC8BB4E8189E
   Responder cookie: F019425F61E3FB68
   Next payload: Key Exchange (4)
   Version: 1.0
   Exchange type: Identity Protection (Main Mode) (2)
   Flags: 0x00
      .... ...0 = Not encrypted
      .... ..0. = No commit
      .... .0.. = No authentication
   Message ID: 0x00000000
   Length: 192
   Key Exchange payload
      Next payload: Nonce (10)
      Payload length: 100
      Key Exchange Data (96 bytes / 768 bits)
   Nonce payload
      Next payload: NAT-D (draft-ietf-ipsec-nat-t-ike-01 to 03) (130)
      Payload length: 24
      Nonce Data
   NAT-D (draft-ietf-ipsec-nat-t-ike-01 to 03) payload
      Next payload: NAT-D (draft-ietf-ipsec-nat-t-ike-01 to 03) (130)
      Payload length: 20
      Hash of address and port: 03C4F5874FBB9FA58ABA4B19906DC59E
   NAT-D (draft-ietf-ipsec-nat-t-ike-01 to 03) payload
      Next payload: NONE (0)
      Payload length: 20
      Hash of address and port: 3091D1A5834CD9B8B59054BA994F5389
```

As each computer had transmitted their nonces, they then entered into the information protection mode where the rest of the transmission was encrypted using the 768-bit MODP group Diffie-Hellman algorithm. Of note, here, is that the vulnerability of a Man-in-the-Middle attack is negated by the use of nonces uniquely generated by each computer by using a private, unique integer.

After the rest of ISAKMP and IKE had finished determining the SAs for the rest of the communication, Authentication Headers were inserted after the IP header as the following example shows:

```
Authentication Header
   Next Header: TCP (0x06)
   Length: 24
   AH SPI: 0x1bb9f567
   AH Sequence: 1
   AH ICV: 246DCADEEFCAFDEE989C1EA6
```

The AH SPI stays the same to identify the Security Payload being utilized by the two computers. The AH Sequence iterates for every pair of packets (i.e. each packet and its reply). The ICV gives the HMAC-MD5 hash utilized by the computers to provide authenticity for the packets.

We then tested the ESP of IPSec with the same transfer of the file via ftp. All of the steps to include the exchanging of nonces were the same. The difference came in that rather than the Authentication header appearing, the entire packet was encrypted, minus the IP header. The following gives an example:

```
Encapsulating Security Payload
   ESP SPI: 0xf31a0c92
   ESP Sequence: 1

0000  00 0d 56 a4 b5 4a 00 0d 56 a5 d8 b7 08 00 45 00   ..V..J..V.....E.
0010  00 50 52 ca 40 00 80 32 12 5e c0 a8 0a 02 c0 a8   .PR.@..2.^......
0020  0a 01 f3 1a 0c 92 00 00 00 01 99 4f a5 89 8c 53   ...........O...S
0030  e7 85 21 b4 d2 3e 02 14 a4 a7 a1 b8 df 81 8f 4b   ..!..>.........K
0040  d6 84 4e aa 33 3f 2d b0 8e 44 2a 06 10 0c 7e 45   ..N.3?-..D*...~E
0050  70 a5 7b 8f ea b5 40 66 bc 41 b8 8d 53 bc         p.{...@f.A..S.
```

The SPI and Sequence for the ESP work the same as for the Authentication Header. What follows, though, is the encryption of the payload using the DES-CBC algorithm.

3.4 IPSec With IPv6 and Linux

Our next attempt was at trying to institute IPSec with IPv6 between the computers. As has been consistent with Windows XP in regards to IPv6 implementation of networking protocols, the user has to utilize the program •IPsec6.exe• from the command line. This gives a broken table layout listing all of the SAs employed by the user on that system. Though allowing for the employment of IPv6 SAs via the uploading of an SA file via IPsec6.exe, there is limited support online for formatting and even fewer examples. Thus secpol.exe only applies to the implementation of IPSec on IPv4. In addition, though we were unable to test, it is documented that encryption in the ESP

packet is not supported for the current implementation of IPv6 IPSec for Windows XP (see http://www.microsoft.com/technet/network/ipv6/ipv6faq.mspx).

We then attempted to implement IPSec on SUSE Linux. The two main programs utilized for this operation are OpenSwan (now known as StrongSwan) and kvpnc. However, the only capability of kvpnc interfacing with OpenSwan was for the connection and establishment of VPN tunnels with other networks. Due to time constraints, we did not modify OpenSwan or attempt any further application of IPSec with Linux.

4. Results

In this research study, we built both a fully functioning IPv4 network and a fully functioning IPv6 network. On the IPv4 network, we then tested and analyzed the packets transferred during the IPSec implementation. We were able to test the ramifications of different settings allowed by Windows XP for the implementation of IPSec to include the use of Authentication Headers and Encapsulating Payloads. We were also able to perform basic network tasks (ping and ftp) across a network using both the AHs and ESPs of IPSec.

After testing the basic network tasks and collecting the packets of the transaction via Wireshark, we were able to verify the process that IPSec implements, the uniqueness of the nonces transferred, the iteration of the Sequence Number, the SPI used to denote the SA established, and either the ICV or encrypted data that was sent. However, we noticed that neither Windows XP nor current Linux flavors offer Encapsulating Security Payloads for IPv6, and that the implementation of AHs with either OS is difficult and requires manual system administration. In addition, external to internal 1:1 Network Address Translation breaks the Authentication Header.

5. Recommendations

Further research in the implementation of IPSec is needed for more current Microsoft Operating Systems to include Windows Vista. Many of the issues that are faced by Windows XP may be due to the age of the OS and its quick retirement to a •legacy system•. As many of the more current operating systems provide better support for IPv6 in general, IPSec might also be better supported. In addition, further research is necessary for the utilization of third-party implementations of IPSec, especially OpenSwan.

Another area that was not tested was the employment of Virtual Private Networks utilizing IPSec. This can afford the ability to test and study the implications of network addressing schemes, especially ones that differ between networks connected via VPNs. In order to effectively test this employment of IPv6, the use of the IPv6/IPv4 tunnel that is already established with the United States Military Academy provides an excellent opportunity. However, our limitation in testing the employment of VPNs was primarily due to the limit of available routers that could support the construction of a VPN.

6. Conclusion

The goals we established from the onset of the research study were threefold: 1) to test and analyze the implementation of IPSec over an IPv4 network, 2) test and analyze the implementation of IPSec over an IPv6 network, and 3) test and analyze the implementation of IPSec over an IPv4/IPv6 network. These were to be tested using common, current operating systems to include Windows XP SP2 and SUSE 10.3 Linux.

The first goal was achieved as we were able to fully implement, test, and analyze the IPSec protocol over an IPv4 network. Microsoft Windows XP SP2 provides a relatively easy console for the administration of IPSec SAs and Policies for the computer and provides a flexibility to allow for interoperability between many other setups. We were not able to test on the Linux machine due to the nature of the implementation of IPSec, namely that of VPNs solely. From the testing of the IPSec packets, we were able to identify and track both the ISAKMP and IKE protocols and the construction of the SAs between two different computers. We also successfully transferred a file using the File Transport Protocol (FTP) successfully employing both AHs and ESPs.

The second goal was not fully achieved as we were unable to implement a fully working test case of IPSec over IPv6. However, due both to documentation as well as our testing, Windows XP can be said to not fully support the IPSec protocol in regards to IPSec. Though not modified prior to compilation, the OpenSwan third-party implementation of IPSec for SUSE also did not provide IPv6 support. As Linux does not have other native employments of IPSec, it can be assumed that current versions of Linux, though natively supporting IPv6, per se, do not fully support IPSec for IPv6. Although we were unable to test the use of IPSec over a network that employs both IPv4 and IPv6, it can be assumed that the difference in headers between the two protocols breaks the ability for IPSec to work, especially as it is employed on the Network Layer. Further thoughts on the general security implications of IPSec are offered in Appendix C.

Therefore, IPSec affords networks many added security features. Rather than having to rely upon system administration for the security of individual applications, all protocols can be safely encrypted on an end-to-end basis. However, IPSec does not work with Network Address Translation. This is particularly disconcerting as the current use of the IPv4 protocol depends entirely upon the employment of NAT for the salvation of its limited address space. Thus it appears that IPSec is better suited for IPv6. However, to date, there is limited support for the implementation of IPSec in IPv6 by current vendors.

Appendix A: Diffie-Hellman and Key Creation

The Diffie-Hellman key exchange algorithm, invented in 1976, was the first public-key algorithm to be developed. The beauty of the Diffie-Hellman algorithm is its openness and resultant simplicity. The strength of the algorithm is derived from the difficulty in calculating discrete logarithms in finite fields. Of note, however, is the fact that that the Diffie-Hellman algorithm cannot be used for encryption or decryption, but rather for key exchange and distribution. A similar algorithm, the ElGamal scheme can be used for both digital signatures and encryption.

The following presents an example of a Diffie-Hellman exchange:

$$\begin{array}{ccc} & g \qquad n & \\ x & \text{<----------} > & y \\ k_x & & k_y \\ g^{k_x} \bmod n & & g^{k_y} \bmod n \\ \left(g^{k_y} \bmod n\right)^{k_x} \bmod n & & \left(g^{k_x} \bmod n\right)^{k_y} \bmod n \\ g^{k_y k_x} \bmod n & \equiv & g^{k_x k_y} \bmod n \end{array}$$

The above gives a basic communication between two parties, in this case *x* and *y*, and how the Diffie-Hellman is derived for their communication. At the start of the communication, *x* and *y* both agree on *n*, a large prime number, and *g*, such that it is a primitive mod *n*, where $\text{mod}(a,n) = a - n \times floor(a/n)$. These two numbers can be passed in the open over an insecure system. After the exchange of *g* and *n*, *x* chooses a random large integer, k_x, and y chooses a random large integer, k_y. These are kept secret by the parties and are not transmitted as is to other parties. When a random large integer has been chosen, *x* then calculates and transmits $g^{k_x} \bmod n$ to *y*. Similarly, *y* calculates and transmits $g^{k_y} \bmod n$ to *x*. Both *x* and *y* then take the values transmitted to them and multiply to the power of their key, their selected random large integer, such that *x* calculates $\left(g^{k_y} \bmod n\right)^{k_x} \bmod n$. This simplifies to $g^{k_y k_x} \bmod n$ for *x* which is equal to the calculation for *y*: $g^{k_x k_y} \bmod n$. Because $g^{k_y k_x} \bmod n \equiv g^{k_x k_y} \bmod n$, both parties can communicate in the open, sharing publicly *g* and *n*, while never allowing for $g^{k_y k_x} \bmod n$ to be calculated by a third party.

The use of the Diffie-Hellman algorithm is utilized by Internet Key Exchange Protocol for the development of a common key between two parties for the construction of both Authentication Headers and Encapsulating Security Payload. The Diffie-Hellman group is determined by the ISAKMP SAs that are established between the two parties and offer, besides different bit and field sizes, the use of either modulus prime or an elliptical curve analog.

One weakness of the use of the Diffie-Hellman algorithm, however, is the vulnerability to Man-in-the-Middle-Attacks. If a person, say *z*, can place themselves between *x* and *y* and perform the Diffie-Hellman on both sides of the communication, then *x* and *y* will not be able to detect an eavesdropper in the communication. This demonstrates the reason for ISAKMP•s use of cookies, specifically the *Photuris* key exchange, and SPIs to identify and designate the communication and actual individuals involved.

Appendix B: Internet Key Exchange and the method of Authentication

For the IKE, each party generates a secret key (SK) and then three other secrets dependent on the SK. These three are the SK_d, used for deriving keying material for IPSec, the SK_a, used for providing data integrity and authentication, and SK_e, used for encrypting IKE messages. Both the initiator and the responder provide their cookies (IC and RC) in addition to their nonce (*Ni* and *Nr*). Of note, the Diffie-Hellman secret generated is also implemented in the *SK* generation. Thus the SK is created in the following manner where •.• denotes concatenation :

For preshared keys:

$$SK = PRF(preshared - key, Ni.Nr)$$

For digital signatures:

$$SK = PRF(Ni.Nr, g^{xy})$$

For nonces:

$$SK = PRF(hash(Ni.Nr), IC.IR)$$

where PRF usually denotes the HMAC-version of the aforementioned pseudo-random function that was negotiated. Once the SK has been derived, the three other secrets are derived:

$$SK_d = PRF(SK, g^{xy}.IC.IR.0)$$
$$SK_a = PRF(SK, SK_d.g^{xy}.IC.IR.1)$$
$$SK_e = PRF(SK, SK_a.g^{xy}.IC.IR.2)$$

If the required number of bits is larger than what is initially given, then the SK_e is expanded via feedback and concatenation in the following manner:

$$Key = Key1.Key2.Key3.Key4$$
$$Key1 = PRF(SK, 0)$$
$$Key2 = PRF(SK, Key1, 1)$$
$$Key3 = PRF(SK, Key1, 2)$$
$$Key4 = PRF(SK, Key2, 3)$$

During Phase 1, exchanges are authenticated by both sides computing unique hashes in the following manner:

Initiator•s Hash:

$$I - HASH = PRF(SK, g^i.g^r.IC.RC.SA.ID_I)$$

Responder•s Hash:

$$R - HASH = PRF(SK, g^r.g^i.RC.IC.SA.ID_R)$$

Appendix C: IPv6 and General Security Implications

In the face of ultimately reaching IP address capacity and the ever-growing security threat posed by the inherent deficiencies in the present IPv4 Internet Protocol version, the world incredulously continues to move slowly towards the only conceivable solution: conversion to Internet Protocol version 6 (IPv6). But universal conversion to IPv6 does not come without a price. Hindrances such as time, cost and inherent conversion complexities thwart the necessary efforts of more well-developed countries such as the US which pragmatically dismisses the conversion as an unnecessary change while many developing countries see it as the only means of building their IT infrastructure where critical IP address capacity is available. Additionally, other factors such as cell-phone technologies which utilize IPv6 are driving the need for this universal conversion.

But besides the greater address capacity, a benefit that is often promoted for IPv6 conversion is that of the mandatory inclusion of IPSec into the new protocol. However, this tends to be shortsighted as the problems that have belabored the implementation of IPSec with IPv4 are the same problems that will be faced with IPv6 (namely key management, configuration complexities, et cetera). An aspect where the IPv6 protocol does provide an innate benefit regarding network security is the lack of Network Address Translation. IPv6 does not suffer from the issue of small subnets as its default subnet size is actually 64 bits. This size greatly hinders both the ability to scan entire subnets and the ability for many worms to propagate at the speeds seen today. In addition, the multiple addresses assigned to a given node (link-local, site-local, global) make spoofing attempts much more transparent and more preventable. This is mainly due to the fact that link-local address is used for infrastructure communication.

However, IPv6 does not just provide possible solutions to the current problems. Because it is a rather new protocol in development, there is a current lack of operational experience as well as software support readily available with IPv4. Many implementations of IPv6 are experimental by nature, and any serious, full-scale network-wide implementation of IPv6 requires the use of Dual-Stacked IPv4 services to make a fully-functioning network. This obviously necessitates the presence of a duplicitous IPv4 network. Therefore, regardless of the growing demand to convert to IPv6, IPv4 by necessity is not going away any time soon. Ironically, it will pose an increasing vulnerability as attention is turned increasingly from IPv4 to IPv6.

Inherent specifically to IPv6, IPSec does not give backwards compatibility. As the headers are starkly different, the end result is the turning off of IPSec features that were actually created to add security. Thus we have to make our networks more insecure to allow for cross-protocol support. In addition, key management plays a huge role in the implementation of IPSec for IPv6, especially due to the large nature of IPv6 networks and the resultant costs involved.

Although IPv6 conversion provides many solutions and also poses some new problems in the issues of network security, many of the fundamentals remain the same regarding the new protocol. IPv6 is not, by its nature, more secure. What will make it more secure is the manner by which it is implemented and the cost-benefit and forward-looking decisions that will have to be made by management in both the private and public sectors.

Appendix D: Lua Source Code Demonstrating Diffie-Hellman

```
--ikea.lua
print'Alice and Bob wish to communicate in such a fashion as to
publicly agree on a key, yet hide their secrets'

print'Alice and Bob agree on a base number (g).  Enter g
(normally 2 or 5):'
g = io.stdin:read'*l'


print'Alice and Bob also agree on a prime number (n). Enter n:'
n = io.stdin:read'*l'

print'Alice wishes to keep a secret (kx). Enter kx:'
kx = io.stdin:read'*l'

print'Bob also wishes to keep a secret (ky). Enter ky:'
ky = io.stdin:read'*l'

print('g= ',g)
print('n= ',n)

gkx = math.pow(g, kx)
gky = math.pow(g, ky)

b = gkx % n
e = gky % n

ekx = math.pow(e, kx)
bky = math.pow(b, ky)

fin1 = ekx % n
fin2 = bky % n

print('kx= ', kx)
print('ky= ', ky)
print('g^kx= ', gkx)
print('g^ky= ', gky)
print('b= ', b)
print('e= ', e)
print('(g^kx mod n)^ky = ',bky)
print('(g^ky mod n)^kx = ',ekx)
print('(g^kx mod n)^ky mod n = ', fin1)
print('(g^ky mod n)^kx mod n = ', fin2)
print('Press any key to exit')
zzend = io.stdin:read'*l'
```

References

[1] S. Convery, *Network Security Architectures*, Cisco Press, 2004.

[2] N. Doraswamy and D. Harkins, *IPSec: the New Security Standard for the Internet, Intranets, and Virtual Private Networks*, 2nd ed. Upper Saddle River: Prentice Hall, 2003.

[3] B. Schneier, *Applied Cryptography*, 2nd ed. John Wiley & Sons, Inc, 1996.

[4] J. Snader, *VPNs Illustrated: Tunnels, VPNs, and IPsec*, Addison-Wesley, 2006.

[5] D. Maughan, *Internet Security Association and Key Management Protocol (ISAKMP)*, IETF RFC 2408, 1998; http://www.faqs.org/rfcs/rfc2408.html.

[6] D. Harkins, *The Internet Key Exchange (IKE)*, IETF RFC 2409, 1998; http://www.faqs.org/rfcs/rfc2409.html.

[7] Microsoft Corporation, •IPv6 for Microsoft Windows: Frequently Asked Questions•, Microsoft Corporation, 2002, http://www.microsoft.com/technet/network/ipv6/ipv6faq.mspx.